Towards a
Complexity-through-Realisability Theory

Thomas Seiller[a,1,*]

[a] *Proofs, Programs, Systems. CNRS – Paris Diderot University.*

Abstract

We explain how recent developments in the fields of Realisability models for linear logic [1] – or *geometry of interaction* – and implicit computational complexity [2, 3] can lead to a new approach of implicit computational complexity. This semantic-based approach should apply uniformly to various computational paradigms, and enable the use of new mathematical methods and tools to attack problem in computational complexity. This paper provides the background, motivations and perspectives of this *complexity-through-Realisability* theory to be developed, and illustrates it with recent results [4].

1. Introduction

Complexity theory lies at the intersection between mathematics and computer science, and studies the amount of resources needed to run a specific program (complexity of an algorithm) or solve a particular problem (complexity of a problem). wewill explain how it is possible to build on recent work in *Realisability models for linear logic* – a mathematical model of programs and their execution – to provide new characterisations of existing complexity classes. It is hoped that these characterisations will enable new mathematical techniques, tools and invariants from the fields of operators algebras and dynamical systems, providing researchers with new tools and methods to attack long-standing open problems in complexity theory.

The *complexity-through-Realisability* theory we propose to develop will provide a unified framework for studying many computational paradigms and their associated computational complexity theory grounded on well-studied mathematical concepts. This should provide a good candidate for a theory of complexity for computational paradigms currently lacking an established theory (e.g. concurrent processes), as well as contribute to establish a unified and well-grounded account of complexity for higher-order functionals.

Even though it has been an established discipline for more than 50 years [5], many questions in complexity theory, even basic ones, remain open. During the last twenty years, researchers have developed new approaches based on logic:

---

they offer solid, machine-independent, foundations and provide new tools and methods. Amongst these approaches, the fields of Descriptive Complexity (DC) and Implicit Computational Complexity (ICC) lead to a number of new characterisations of complexity classes. These works laid grounds for both theoretical results [6] and applications such as typing systems for complexity constrained programs and type inference algorithms for statically determining complexity bounds.

The *complexity-through-Realisability* theory we propose to develop is related to those established logic-based approaches. As such, it inherits their strengths: it is machine-independent, provides tools and methods from logic and gives grounds for the above mentioned applications. Furthermore, it builds on state-of-the-art theoretical results on Realisability models for linear logic [1] using well-studied mathematical concepts from operators algebras and dynamical systems. As a consequence, it opens the way to use against complexity theory's open problems the many techniques, tools and invariants that were developed in these disciplines.

We illustrate the approach by explaining how first results were recently obtained by capturing a large family of complexity classes corresponding to various notions of automata. Indeed, we provided [4] Realisability models in which types of binary predicates correspond to the classes of languages accepted by one-way (resp. two-way) deterministic (resp. non-deterministic, resp. probabilistic) multi-head automata. This large family of languages contains in particular the classes REGULAR (regular languages), STOCHASTIC (stochastic languages), LOGSPACE (logarithmic space), NLOGSPACE (non-deterministic logarithmic space), CON-LOGSPACE (complementaries of languages in NLOGSPACE), and PLOGSPACE (probabilistic logarithmic space).

Finally, we discuss the possible extensions and further characterisations that could be obtained by the same methods. We in particular discuss the relationship between the classification of complexity classes and the problem of classifying a certain kind of algebras, namely *graphing algebras*.

1.1. Complexity Theory

Complexity theory is concerned with the study of how many resources are needed to perform a specific computation or to solve a given problem. The study of *complexity classes* – sets of problems which need a comparable amount of resources to be solved, lies at the intersection of mathematics and computer science. Although a very active and established field for more than fifty years [7, 8, 9, 10], a number of basic problems remain open, for instance the famous "millennium problem" of whether PTIME equals NPTIME or the less publicized but equally important question of whether PTIME equals LOGSPACE. In recent years, several results have greatly modified the landscape of complexity theory by showing that proofs of separation (i.e. inequality) of complexity classes are hard to come by, pointing out the need to develop new theoretical methods. The most celebrated result in this direction [11] defines a notion of *natural proof* comprising all previously developed proof methods and shows that no "natural proof" can succeed in proving separation.

Mathematicians have then tried to give characterisations of complexity classes that differ from the original machine-bound definitions, hoping to enable methods from radically different areas of mathematics. Efforts in this direction lead to the development of Descriptive Complexity (DC), a field which studies the types

of logics whose individual sentences characterise exactly particular complexity classes. Early developments were the 1974 Fagin-Jones-Selman results [12, 13] characterising the classes NExptime and NPtime. Many such characterisations have then been given [14, 15, 16] and the method led Immerman to a proof of the celebrated Immerman-Szelepcsényi theorem [6, 17] stating the two complexity classes coNLogspace and NLogspace are equal (though Szelepcsényi's proof does not use logic-based methods).

Implicit Computational Complexity (ICC) develops a related approach whose aim is to study algorithmic complexity only in terms of restrictions of languages and computational principles. It has been established since Bellantoni and Cook' landmark paper [18], and following work by Leivant and Marion [19, 20]. Amongst the different approaches to ICC, several results were obtained by considering syntactic restrictions of *linear logic* [21], a refinement of intuitionnistic logic which accounts for the notion of resources. Linear logic introduces a modality ! marking the "possibility of duplicating" a formula A: the formula A shall be used exactly once, while the formula $!A$ can be used any number of times. Modifying the rules governing this modality then yields variants of linear logic having computational interest: this is how constrained linear logic systems, for instance BLL [22] and ELL [23], are obtained. However, only a limited number of complexity classes were characterised in this way, and the method seems to be limited by its syntactic aspect: while it is easy to modify existing rules, it is much harder to find new, alternative, rules from scratch. The approach we propose to follow in this paper does not suffer from these limitations, allowing for subtle distinctions unavailable to the syntactic techniques of ICC.

1.2. Realisability Models for Linear Logic

Concurrently to these developments in computational complexity, and motivated by disjoint questions and interests, Girard initiated the Geometry of Interaction (GoI) program [24]. This research program aims at obtaining particular kinds of *Realisability* models (called GoI models) for linear logic. Realisability was first introduced [25] as a way of making the Brouwer-Heyting-Kolmogorov interpretation of constructivism and intuitionistic mathematics precise; the techniques were then extended to classical logic, for instance by Krivine [26], and linear logic. The GoI program quickly arose as a natural and well-suited tool for the study of computational complexity. Using the first GoI model [27], Abadi, Gonthier and Lévy [28] showed the optimality of Lamping's reduction in lambda-calculus [29]. It was also applied in implicit computational complexity [30], and was the main inspiration behind dal Lago's context semantics [31].

More recently the geometry of interaction program inspired new techniques in implicit computational complexity. These new methods were initiated by Girard [32] and have known a rapid development. They lead to a series of results in the form of new characterisations of the classes coNLogspace [32, 2], Logspace [3, 33] and Ptime [34]. Unfortunately, although the construction of Realisability models and the characterisations of classes are founded on similar techniques, they are two distinct, unrelated, constructions. The approach we propose to develop here will in particular bridge this gap and provide similar characterisations which will moreover allow the use of both logical and Realisability-specific methods.

2. A *Complexity-through-Realisability* Theory

2.1. Technical Background and Motivations

About ten years ago, Girard showed [35] that the restriction to the unit ball of a von Neumann algebra of the so-called "feedback equation", which represents the execution of programs in GoI models, always has a solution. Moreover, previous and subsequent work showed the obtained GoI model interprets, depending on the choice of the von Neumann algebra, either full linear logic [36] or the constrained system ELL which characterises elementary time computable functions [37]. This naturally leads to the informal conjecture that there should be a correspondence between von Neumann algebras and complexity constraints.

This deep and promising idea turned out to be slightly inexact and seemingly difficult to exploit. Indeed, the author showed [37, 38] that the expressivity of the logic interpreted in a GoI model depends not only on the enveloping von Neumann algebra $\mathfrak{N}$ but also on a maximal abelian sub-algebra (masa) $\mathfrak{A}$ of $\mathfrak{N}$, hinting at a refined conjecture stating that complexity constraints correspond to such couples $(\mathfrak{A}, \mathfrak{N})$. This approach is however difficult to extend to exploit and adapt to model other constrained logical systems for two reasons. The first reason is that the theory of maximal abelian sub-algebras in von Neumann algebras is an involved subject matter still containing large numbers of basic but difficult open problems [39]. The second is that even though some results were obtained, no intuitions were gained about what makes the correspondence between couples $(\mathfrak{A}, \mathfrak{N})$ and complexity constraints work.

Some very recent work of the author provides the foundational grounds for a new, tractable way of exploring the latter refined conjecture. This series of work [40, 41, 1, 42] describes a systematic construction of realisability models for linear logic which unifies and extends all GoI models introduced by Girard. The construction is built upon a generalization of graphs, named *graphings* [43, 44, 45], which can be understood either as *geometric realisations* of graphs on a measure space $(X, \mathscr{B}, \mu)$, as measurable families of graphs, or as generalized measured dynamical system. It is parametrized by two monoids describing the model of computation and a map describing the realisability structure:

- a monoid Ω used to associate weights to edges of the graphs;
- a map $m : \Omega \to \bar{\mathbf{R}}_{\geqslant 0}$ defining *orthogonality* – accounting for linear negation;
- a monoid $\mathfrak{m}$ – the *microcosm* – of measurable maps from $(X, \mathscr{B}, \mu)$ to itself.

A Ω-*weighted graphing in* $\mathfrak{m}$ is then defined as a directed graph F whose edges are weighted by elements in Ω, whose vertices are measurable subsets of the measurable space $(X, \mathscr{B})$, and whose edges are *realised* by elements of $\mathfrak{m}$, i.e. for each edge e there exists an element ϕ_e in $\mathfrak{m}$ such that $\phi_e(s(e)) = t(e)$, where s, t denote the source and target maps. Based on this notion, and an orthogonality relation defined from the map m, we obtained a systematic method for constructing realisability models for linear logic.

Theorem 1 (Seiller [1])**.** *For all choices of Ω, $\mathfrak{m}$ and $m : \Omega \to \bar{\mathbf{R}}_{\geqslant 0}$, the set of Ω-weighted graphings in $\mathfrak{m}$ defines a model of Multiplicative-Additive Linear Logic (*MALL*).*

Let us notice that a microcosm $\mathfrak{m}$ which contains only measure-preserving maps generates a measurable equivalence relation which, by the Feldman-Moore

construction [46], induces a couple $(\mathfrak{A}, \mathfrak{N})$ where $\mathfrak{A}$ is a maximal abelian subalgebra of the von Neumann algebra $\mathfrak{N}$. The previous theorem thus greatly generalizes Girard's general solution to the feedback equation [35], and in several ways. First, we define several models (deterministic, probabilistic, non-deterministic), among which some are unavailable to Girard's approach: for instance the non-deterministic model violates Girard's norm condition, and more generally Girard's techniques only apply when Ω is chosen as a subset of the complex plane unit disc. It shows in fact much more as it exhibits an infinite family of structures of realisability models (parametrized by the map $m : \Omega \to \mathbf{R}_{\geqslant 0} \cup \{\infty\}$) on any model obtained from a microcosm. This extends drastically Girard's approach for which only two such structures were defined until now: an "orthogonality-as-nilpotency" structure in the algebra $\mathscr{L}(\mathbb{H})$ [27] and another one defined using the Fuglede-Kadison determinant [47] in the type II_1 hyperfinite factor [48].

2.2. Methodology

We can now explain the proposed methodology for defining a *Complexity-through-realisability* theory.

The notion of Ω-weighted $\mathfrak{m}$-graphings for given monoid Ω and microcosm $\mathfrak{m}$ yields a very general yet tractable mathematical notion of algorithm, as the microcosm $\mathfrak{m}$ can naturally be understood as a set of computational principles [49]. It therefore provides an interesting middle-ground between usual computational models, for instance automata, and mathematical techniques from operator algebras and dynamical systems. It comprises Danos' interpretation of pure lambda-calculus (a Turing complete model of computation) in terms of operators [50], but it is not restricted to sequential algorithms as it will be shown to provide an interpretation of probabilistic and quantum programs. It also provides characterisations of usual complexity classes as types of predicates over binary words $\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Bool}$, which will lead to a partial proof of the above conjecture by showing a correspondence between families of microcosms and complexity constraints.

Work in this direction will establish these definitions of algorithms and complexity constraints as a uniform, homogeneous, machine-independent approach to complexity theory. The methods developed in this setting, either adapted from DC/ICC or specific to the realisability techniques employed, will apply to probabilistic/quantum complexity classes as much as sequential classes. In particular, it will offer a framework where comparison between non-classical and classical classes can be performed. It will also expand to computational paradigms where no established theory of complexity exists, providing a strong and coherent proposition for such.

It will extend the approach of ICC and DC as it will go beyond the syntactical restrictions they are suffering from. In particular, it will provide a new method for defining logical systems corresponding to complexity classes: the realisability model construction gives a systematic way to define a logic corresponding to the underlying computational model. It will also extend the GoI model approach to complexity by reconciling the logical and complexity aspects, allowing the use of both logical and realisability-specific methods.

Lastly, the approach we propose to develop does not naturally fall into the usual pitfalls for the obtention of separation results. Therefore, it provides a framework which will potentially offer separation methods, e.g. using invariants for the well-established mathematical notions it is founded upon.

3. Interaction Graphs Models of Linear Logic

3.1. Graphings

Definition 2. Let $(X, \mathscr{B}, \lambda)$ be a measure space. We denote by $\mathscr{M}(X)$ the set of non-singular transformations[2] $X \to X$. A *microcosm* of the measure space X is a subset $\mathfrak{m}$ of $\mathscr{M}(X)$ which is closed under composition and contains the identity.

In the following, we will consider a notion of graphing depending on a *weight-monoid* Ω, i.e. a monoid $(\Omega, \cdot, 1)$ which contains the possible weights of the edges.

Definition 3 (Graphings)**.** Let $\mathfrak{m}$ be a microcosm of a measure space $(X, \mathscr{B}, \lambda)$ and V^F a measurable subset of X. A Ω-*weighted graphing in* $\mathfrak{m}$ *of carrier* V^F is a countable family $F = \{(\omega_e^F, \phi_e^F : S_e^F \to T_e^F)\}_{e \in E^F}$, where, for all $e \in E^F$ (the set of *edges*):
- ω_e^F is an element of Ω, the *weight* of the edge e;
- $S_e^F \subset V^F$ is a measurable set, the *source* of the edge e;
- $T_e^F = \phi_e^F(S_e^F) \subset V^F$ is a measurable set, the *target* of the edge e;
- ϕ_e^F is the restriction of an element of $\mathfrak{m}$ to S_e^F, the *realisation* of the edge e.

It was shown in earlier work [41] how one can construct models of MALL where proofs are interpreted as graphs. This construction relied on a single property, called the *trefoil property*, which relates two simple notions:
- the *execution* $F :: G$ of two graphs, a graph defined as a set of paths;
- the *measurement* $[\![F, G]\!]_m$, a real number computed from a set of cycles.

These constructions can be extended to the more general framework where proofs are interpreted as graphings. Indeed, the notions of paths and cycles in a graphings are quite natural, and from two graphings F, G in a microcosm $\mathfrak{m}$ one can define its execution $F :: G$ which is again a graphing in $\mathfrak{m}$[3]. A more involved argument then shows that the trefoil property holds for a family of measurements $[\![\cdot, \cdot]\!]_m$, where $m : \Omega \to \mathbf{R}_{\geqslant 0} \cup \{\infty\}$ is any measurable map. These results are obtained as a generalization of constructions considered in the author's PhD thesis[4] [37].

Theorem 4 (Enveloping model)**.** *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}[\Omega, \mathfrak{m}]$, of MALL.*

In most of the models, one can define some exponential connectives. In particular, all models considered later on have the necessary structure to define an exponential modality. Let us notice however that the notion of exponential modality we are considering here is extremely weak, as most models won't validate the functorial promotion rule. The only rule that is assured to be satisfied by the exponential connectives we will consider is the contraction rule, i.e. for any type **A**, one has $!\mathbf{A} \multimap !\mathbf{A} \otimes !\mathbf{A}$. These very weak exponential connectives will turn out to be of great interest: we obtain in this way models of linear logic where the exponentials are weaker than what is obtained through syntactic consideration in systems like BLL, SLL, etc. and characterise low complexity classes.

---

[2]A non-singular transformation $f : X \to X$ is a measurable map which preserves the sets of null measure, i.e. $\lambda(f(A)) = 0$ if and only if $\lambda(A) = 0$.

[3]As a consequence, a microcosm characterises a "closed world" for the execution of programs.

[4]In the cited work, the results were stated in the particular case of the microcosm of measure-preserving maps on the real line.

3.2. Models of Computation

Before explaining how one can characterise complexity classes in this way, we need to state refinements of the previous theorem. We first define the notion of *deterministic graphing*.

Definition 5 (Deterministic graphings). A Ω-weighted graphing G is *deterministic* when:
- for all $e \in E^G$, $\omega_e^G \in \{0, 1\}$;
- the following set is of null measure: $\{x \in \mathbf{X} \mid \exists e \neq e' \in E^G, x \in S_e^G \cap S_{e'}^G\}$

Non-deterministic graphings are defined as those graphings satisfying the first condition.

We then prove that the notions of deterministic and non-deterministic graphings are closed under composition, i.e. if F, G are deterministic graphings, then their execution $F :: G$ is again a deterministic graphing. This shows that the sets of deterministic and non-deterministic graphings define submodels of $\mathbb{M}[\Omega, \mathfrak{m}]$.

Theorem 6 (Deterministic model). *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted deterministic graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{det}}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic.*

Theorem 7 (Non-deterministic model). *The set of Ω-weighted non-deterministic graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{ndet}}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic.*

One can also consider several other classes of graphings. We explain here the simplest non-classical model one could consider, namely that of *probabilistic graphings*. In order for this notion to be of real interest, one should suppose that the unit interval $[0, 1]$ endowed with multiplication is a submonoid of Ω.

Definition 8 (Probabilistic graphings). A Ω-weighted graphing G is *probabilistic* when:
- for all $e \in E^G$, $\omega_e^G \in [0, 1]$;
- the following set is of null measure: $\{x \in \mathbf{X} \mid \sum_{e \in E^G, \ x \in S_e^G} \omega_e^G > 1\}$

It turns out that this notion of graphing also behaves well under composition, i.e. there exists a *probabilistic* submodel of $\mathbb{M}[\Omega, \mathfrak{m}]$, namely the model of *probabilistic graphings*.

Theorem 9 (Probabilistic model). *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted probabilistic graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{prob}}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic.*

These models are all submodels of the single model $\mathbb{M}[\Omega, \mathfrak{m}]$. Moreover, other inclusions of models can be obtained by modifying the other parameters, namely the weight monoid Ω and the microcosm $\mathfrak{m}$. For instance, given two microcosms $\mathfrak{m} \subset \mathfrak{n}$, it is clear that a graphing in $\mathfrak{m}$ is in particular a graphing in $\mathfrak{n}$. This inclusion actually extends to an embedding of the model $\mathbb{M}[\Omega, \mathfrak{m}]$ into $\mathbb{M}[\Omega, \mathfrak{n}]$ which preserves most logical operations[5]. Moreover, given two microcosms $\mathfrak{m}$ and $\mathfrak{n}$, one can define the *smallest common extension* $\mathfrak{m} + \mathfrak{n}$ as the compositional closure

---

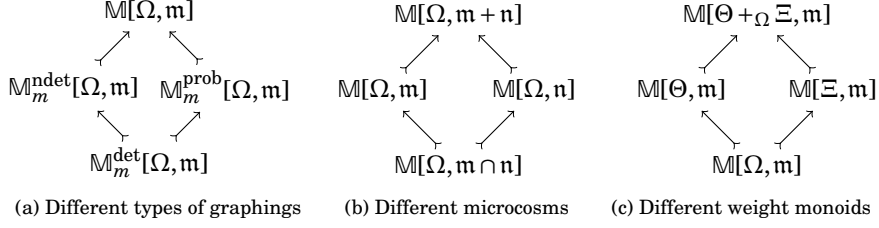[5]It preserves all connectives except for negation.

(a) Different types of graphings   (b) Different microcosms   (c) Different weight monoids

Figure 1: Inclusions of models

of the set $\mathfrak{m} \cup \mathfrak{n}$. The model $\mathbb{M}[\Omega, \mathfrak{m} + \mathfrak{n}]$ then contains both models $\mathbb{M}[\Omega, \mathfrak{m}]$ and $\mathbb{M}[\Omega, \mathfrak{n}]$ through the embedding just mentioned. In the same way, an inclusion of monoids $\Omega \subset \Gamma$ yields an embedding of the the model $\mathbb{M}[\Omega, \mathfrak{m}]$ into $\mathbb{M}[\Gamma, \mathfrak{m}]$. For instance, the model $\mathbb{M}[\{1\}, \mathfrak{m}]$ is a submodel of $\mathbb{M}[\Omega, \mathfrak{m}]$ for any monoid Ω. One can also define, given weight monoids Ω, Θ and Ξ with monomorphisms $\Omega \to \Theta$ and $\Omega \to \Xi$, the model $\mathbb{M}[\Theta +_\Omega \Xi, \mathfrak{m}]$ where $\Theta +_\Omega \Xi$ denotes the amalgamated sum of the monoids. Figure 1 illustrates some of these inclusions of models.

4. characterisations of sub-logarithmic classes

We now expose some recent results obtained by applying the methodology described above [4]. We describe in this way a number of new characterisations of sublogarithmic complexity classes. Before going into details about these characterisations, let us define a number of complexity classes – all of them definable by classes of automata.

Definition 10. For each integer i, we define:
- the class 2DFA(i) (resp. 1DFA(i)) as the set of languages accepted by deterministic two-way (resp. one-way) multihead automata with at most i heads;
- the class 2NFA(i) (resp. 1NFA(i)) as the set of languages accepted by two-way (resp. one-way) multihead automata with at most i heads;
- the class CO2NFA(i) (resp. CO1NFA(i)) as the set of languages whose complementary language is accepted by two-way (resp. one-way) multihead automata with at most i heads;
- the class 2PFA(i) (resp. 1PFA(i)) as the set of languages accepted by two-way (resp. one-way) probabilistic multihead automata with at most i heads;

We also denote by LOGSPACE (resp. PTIME) the class of predicates over binary words that are recognized by a Turing machine using logarithmic space (resp. polynomial time), by NLOGSPACE (resp. NPTIME) its non-deterministic analogue, by CONLOGSPACE (resp. CONPTIME) the set of languages whose complementary language lies in LOGSPACE (resp. PTIME). We also denote by PLOGSPACE the class of predicates over binary words that are recognized by a probabilistic Turing machine with unbounded error using logarithmic space.

We don't recall the usual definitions of these variants of multihead automata, which can be easily found in the literature. We only recall the classical results:

$$\cup_{i \in \mathbf{N}} 2\mathrm{DFA}(i) = \mathrm{LOGSPACE} \qquad \cup_{i \in \mathbf{N}} 2\mathrm{NFA}(i) = \mathrm{NLOGSPACE}$$
$$\cup_{i \in \mathbf{N}} \mathrm{CO2NFA}(i) = \mathrm{CONLOGSPACE} \qquad \cup_{i \in \mathbf{N}} 2\mathrm{PFA}(i) = \mathrm{PLOGSPACE}$$
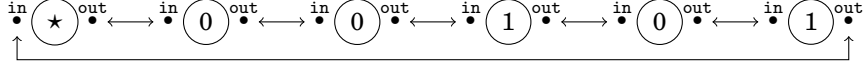
8

Figure 2: Representation of the word $\mathfrak{w} = 00101$

In the following, we will denote by $2\text{DFA}(\infty)$ (resp. $2\text{NFA}(\infty)$, resp. $2\text{PFA}(\infty)$) the set $\cup_{i\in\mathbf{N}}2\text{DFA}(i)$ (resp. $\cup_{i\in\mathbf{N}}2\text{NFA}(i)$, resp. $\cup_{i\in\mathbf{N}}2\text{PFA}(i)$).

4.1. Deterministic Computation: From Regular Languages to Logarithmic Space

In the models of linear logic we described, one can easily define the type $\mathbf{Words}^{(2)}_{\Sigma}$ of words over an arbitrary finite alphabet Σ. The definition of the representation of these binary words comes from the encoding of binary lists in lambda-calculus and is explained thoroughly in previous works [37, 2, 3]. We won't give the formal definition of what is a representation of a word $\mathfrak{w}$ here, but let us sketch the main ideas. Given a word, say the binary word $\mathfrak{w} = 00101$, we introduce a symbol $\star$ that can be understood as a left-hand end-of-tape marker and consider the list of symbols $\star00101$. Then, the graphing that will represent $\mathfrak{w}$ is obtained as a realisation of the directed graph whose set of vertices is $\{\star, 0, 1\} \times \{\texttt{in}, \texttt{out}\}$ and whose edges link the symbols of the list together, i.e. the graph pictured in Figure 2.

We are now interested in the elements of the type $!\mathbf{Words}^{(2)}_{\{0,1\}}$. For each word $\mathfrak{w}$, there exists an element $!L_{\mathfrak{w}}$ in the type $!\mathbf{Words}^{(2)}_{\{0,1\}}$ which represents it. We say that a graphing – or *program* – P of type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ *accepts* the word $\mathfrak{w}$ when the execution $P::W_{\mathfrak{w}}$ is equal to the distinguished element $\texttt{true} \in \mathbf{Bool}$. The *language* accepted by such a program P is then defined as $[P] = \{\mathfrak{w} \in \mathbf{Words}^{(2)}_{\{0,1\}} \mid \phi::W_{\mathfrak{w}} = \texttt{true}\}$.

Definition 11 (characterisation - deterministic models)**.** Let Ω be a monoid, $\mathfrak{m}$ a microcosm and $\mathscr{L}$ a set of languages. We say the model $\mathbb{M}^{\text{det}}[m, \Omega]$ *characterises the set* $\mathscr{L}$ if the set $\{[P] \mid P \in !\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}\}$ is equal to $\mathscr{L}$.

We now consider the measure space $\mathbf{Z} \times [0,1]^{\mathbf{N}}$ endowed with the product of the counting measure on $\mathbf{Z}$ and the Lebesgue measure on the Hilbert cube $[0,1]^{\mathbf{N}}$. To define microcosms, we use the constructor $+$: if $\mathfrak{m}$ and $\mathfrak{n}$ are two microcosms, $\mathfrak{m} + \mathfrak{n}$ is the smallest microcosm containing both $\mathfrak{m}$ and $\mathfrak{n}$. We can now define the following microcosms:

- $\mathfrak{m}_1$ is the monoid of translations $\tau_k : (n, x) \mapsto (n + k, x)$;
- $\mathfrak{m}_{i+1}$ is the monoid $\mathfrak{m}_i + \mathfrak{s}_{i+1}$ where $\mathfrak{s}_{i+1}$ is the monoid generated by the single map:

$$s_{i+1} : (n, (x_1, x_2, \dots)) \mapsto (n, (x_{i+1}, x_2, \dots, x_i, x_1, x_{i+2}, \dots))$$

- $\mathfrak{m}_{\infty} = \cup_{i\in\mathbf{N}}\mathfrak{m}_i$.

The intuition is that a microcosm $\mathfrak{m}$ represents the set of computational principles available to write programs in the model. The operation $+$ thus extends the set of principles at disposal, increasing expressivity. As a consequence, the set of languages characterised by the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ becomes larger and larger as we consider extensions of the microcosms. As an example, the microcosm $\mathfrak{m}_1$ corresponds to allowing oneself to compute with automata. Expanding

9

this microcosm by adding a map s_2 yields $\mathfrak{m}_2 = \mathfrak{m}_1 + \mathfrak{s}_2$ and corresponds to the addition of a new computational principle: using a second head.

Theorem 12. *The model* $\mathbb{M}_m^{\mathrm{det}}[\{1\}, \mathfrak{m}_i]$ ($i \in \mathbf{N} \cup \{\infty\}$) *characterises the class* $2\mathrm{DFA}(i)$.

In particular, the model $\mathbb{M}_m^{\mathrm{det}}[\{1\}, \mathfrak{m}_1]$ characterises the class REGULAR of Regular languages and the model $\mathbb{M}_m^{\mathrm{det}}[\{1\}, \mathfrak{m}_\infty]$ characterises the class LOGSPACE.

4.2. Examples

Examples of words. We here work two examples to illustrate how the representation of computation by graphings works out. First, we give the representation as graphings of the lists $\star 0$ (Figure 3a), $\star 11$ (Figure 3b) and $\star 01$ (Figure 3c). In the illustrations, the vertices, e.g. '0i', '0o', represent disjoint segments of unit length, e.g. $[0, 1]$, $[1, 2]$. As mentioned in the caption, the plain edges are realised as translations. Notice that those edges are cut into pieces (2 pieces each for the list $\star 0$, and three pieces each for the others). This is due to the fact that these graphings represent exponentials of the word representation: the exponential splits the unit segment into as many pieces as the length of the list; intuitively each piece of the unit segments correspond to the "address" of each bit. The edges then explicit the ordering: each symbol comes with has pointers to its preceding and succeeding symbols. E.g. in the case of the first '1' in the list $\star 11$, there is an edge from '1i' to '$\star$o' representing the fact that the preceding symbol was '$\star$', and there is an edge from '1o' to '1i' representing the fact that the next symbol is a '1'; notice moreover that these edges move between the corresponding addresses.

A first machine. As a first example, let us represent an automata that accepts a word $\mathfrak{w}$ if and only if $\mathfrak{w}$ contains at least one symbol '1'. This example was chosen because of the simplicity of the representation. Indeed, one can represent it by a graphing with a single state and which uses a single head (hence it is a graphing in the microcosm $\mathfrak{m}_1$). Its transition relation is then defined as follows: if the symbol read is a '1' then stop, if it is a '0', move the head to read the next symbol, if it is a '$\star$' then reject. The representation as a graphing is shown in Figure 3d. Notice that the symbol '$\star$' plays both the role of left and right end-markers.

We then represent the computation of this graphing with the representations of the words $\star 0$ (Figure 3a), $\star 11$ (Figure 3b) and $\star 01$ (Figure 3c) in Figure 4. The computation is illustrated by showing the two graphings (the machine and the integer), one on each side of the set of vertices. Notice that in those figures the graphing representing the machine has been replaced by one of its *refinements* to reflect the splitting of the unit intervals appearing in the representation of the words. The result of the computation is the set of alternating paths from the set of vertices {accept, reject} to itself: in each case there is at most one, which is stressed by drawing the edges it is composed in boldface. The machine accepts the input if and only if there is a single path from "accept" to "accept". One can see in the figure that this machine accepts the words $\star 11$ and $\star 01$ but not the word $\star 0$.

Notice how the computation can be understood as a game between two players. We illustrate this on the computation on the word $\star 01$. First the machine, through its edge from "accept" to '$\star$o', asks the input "What's your first symbol?". The integer, through its edge from '$\star$o' to '0i', answers "It's a '0'.". Then the machine asks "What's your next symbol?" (the edge from '0i' to '0o'), to which the integer replies "It's a '1'." (the edge from '0o' to '1i'). At this point, the machine accepts (the edge from '1i' to "accept").

A second machine. We chose as a second example a more complex machine. The language it accepts is actually a regular language (the words that contain at least one symbol '1' and one symbol '0'), i.e. can be computed by a single-head automata. However, we chose to compute this language by first looking for a symbol '1' in the word and, if this part is successful, activate a second head looking for a symbol '0', i.e. we compute it with a graphing in the microcosm $\mathfrak{m}_2$ even if as a regular language it can be computed by graphings in $\mathfrak{m}_1$. This automata has two states corresponding to the two parts in the algorithmic procedure: a state "looking for a '1'" and a state "looking for a '0'"; we write these states "L1" and "L0" respectively. The graphing representing this automata is represented in Figure 3e, where there are two sets of vertices one above the other: the row below shows the vertices in state L1 while the row on top shows the vertices in state L2. Notice the two dashed lines which are realised using a permutation and correspond to the change of principal head[6].

We represented the computation of this machine with the same inputs $\star0$, $\star11$, and $\star01$ in Figure 5. Notice that to deal with states, the inputs are duplicated. Once again, acceptance corresponds with the existence of a path from the "accept" vertex to itself (without changing states). The first computation does not go far: the first head moves along the word and then stops as it did not encounter a symbol '1' (this path is shown by thick edges).

The second computation is more interesting. First, it follows the computation of the previous machine on the same input: it asks for the first symbol, acknowledge that it is a '1', then activates the second head and changes state. At this point the path continues as the second head moves along the input: this seems to contradict the picture since we our path seems to arrive on the middle splitting of the vertex '$\star$o' which would not allow us to continue through the input edge whose source is the left-hand splitting of this same vertex. However, this is forgetting that we changed the active head. Indeed, what happens in fact is that the permutation moves the splitting along a new direction: while the same transition realised as a simple translation would forbid us to continue the computation, realising it by the permutation s_2 allows the computation to continue. We illustrate in Figure 6 how the permutation actually interact with the splitting, in the case the interval is split into two pieces. The representation of vertices as unit intervals is no longer sound as we are actually using the two first copies of $[0,1]$ in the Hilbert cube $[0,1]^{\mathbf{N}}$ to represent this computation, hence working with unit squares. In this case, however, the second head moves along the input without encountering a symbol '0' and then stops. This path is shown by thick edges in the figure.

The last computation accepts the input. As in the previous case, the first head moves along the input, encounters a symbol '1', changes its state and activates the second head. As in the previous case, the computation can continue at this point, and the second head encounters a '0' as the first symbol of the input. Then the machine accepts. This path is shown in the figure by thick edges.

---

[6]The machine represented by graphings have a single active head at a given time. The permutations s_i ($i \leqslant 2$) swap the i-th head with the first one, making the i-th active. Reusing the permutation s_i then activates the former active head and "deactivates" the i-th head.

(a) The graphing representing the word ⋆0.

(b) The graphing representing the word ⋆11.

(c) The graphing representing the word ⋆01.

(d) The graphing representing the '1'-machine.
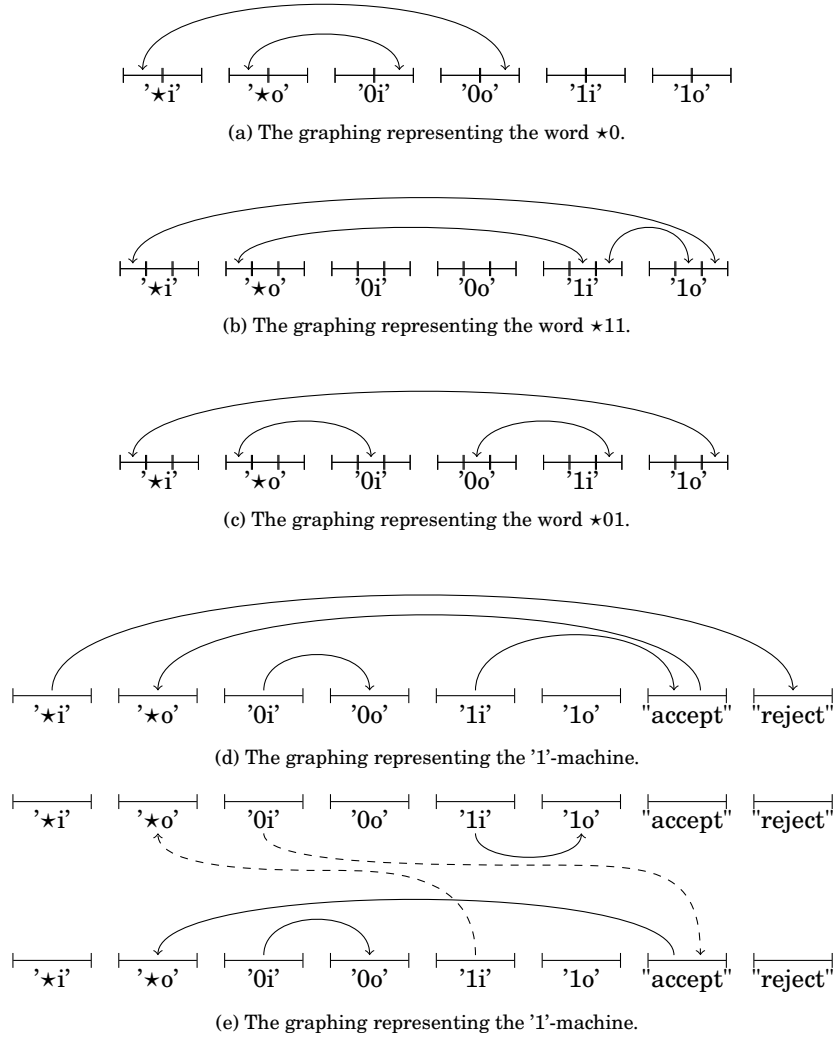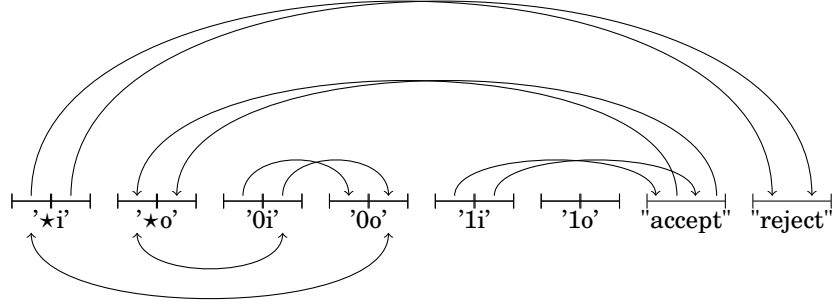

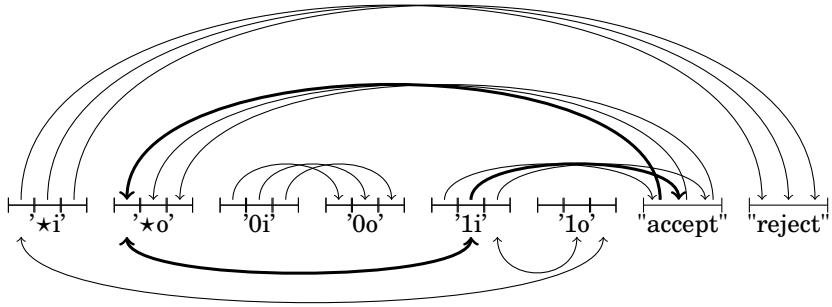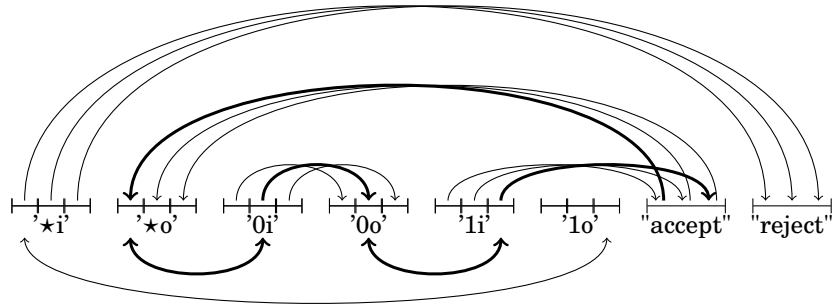(e) The graphing representing the '1'-machine.

Figure 3: Examples: the plain lines are realised by simple translations, while the dashed lines are realised by a composition of a translation and the map s_2.
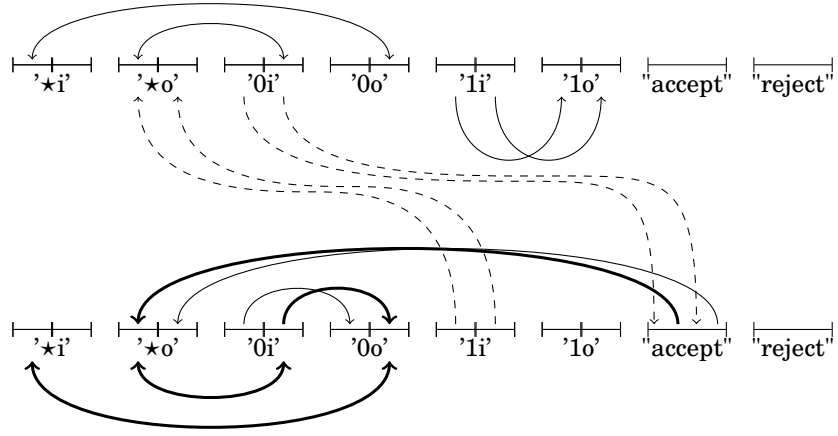
(a) Computing with input ⋆0.

(b) Computing with input ⋆11.
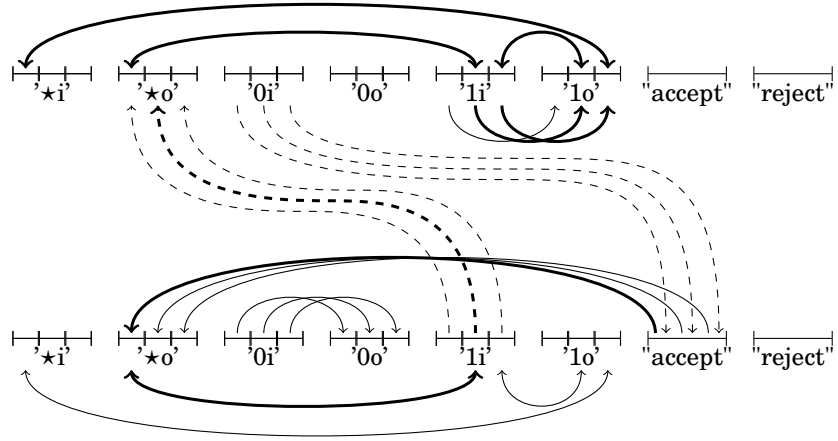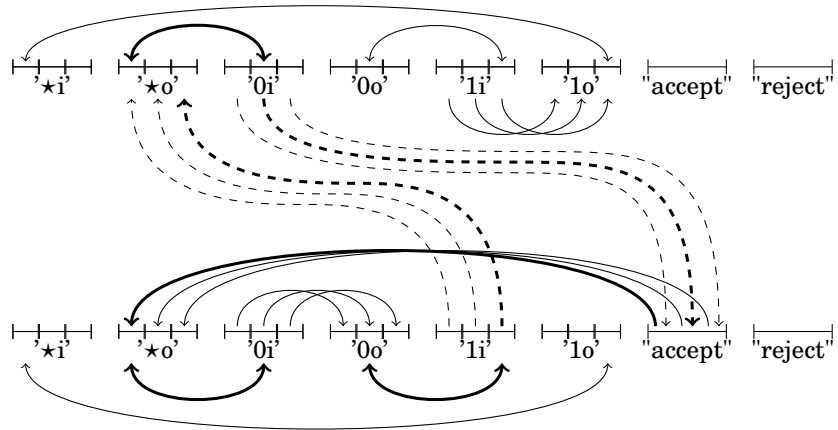

(c) Computing with input ⋆01.

Figure 4: Example: computing with the first machine.

13

(a) Computing with input ⋆0.

(b) Computing with input ⋆11.

(c) Computing with input ⋆01.

Figure 5: Example: computing with the second machine.

(a) A simple translation
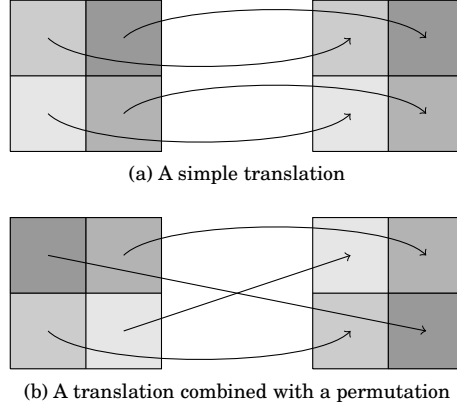

(b) A translation combined with a permutation

Figure 6: How permutations interact with splittings.

4.3. Non-deterministic and Probabilistic Computation

All the preceding results have non-deterministic and probabilistic analogues; we consider in this section the model of non-deterministic graphings. To obtain the same types of results in that case, two issues should be dealt with. First one needs to consider programs of a different type since the result of a non-deterministic computation yield sets of booleans and not a single boolean. Thus, programs will be considered as elements of a more general type than in the deterministic case. We consider here elements of the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}$, where $\mathbf{NBool}$ is a specific type definable in the models, somehow a non-deterministic version of the booleans.

The second issue concerns acceptance. While it is natural in the deterministic case to ask whether the computation yielded the element $\mathtt{true} \in \mathbf{Bool}$, this is no longer the case. Should one define acceptance as producing at least one element $\mathtt{true}$ or as producing no element $\mathtt{false}$? Both conditions should be considered. In order to obtain a quite general notion of acceptance that can not only capture both notions explained above but extend to other computational paradigms such as probabilistic computation, we use the structure of the realisability models we are working with to define a notion of *test*. Indeed, the models are constructed around an orthogonality relation $\perp$: a test will be an element (or more generally a family of elements) $\mathcal{T}$ of the model and a program P accepts the word $\mathfrak{w}$ if the execution $P :: !L_{\mathfrak{w}}$ is orthogonal to $\mathcal{T}$.

Given P an element of $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}$ and a test $\mathcal{T}$, one can define the language $[P]_{\mathcal{T}}$ as the set of all words $\mathfrak{w}$ that are accepted by P w.r.t. the test $\mathcal{T}$:

$$[P]_{\mathcal{T}} = \{\mathfrak{w} \mid P :: !L_{\mathfrak{w}} \perp \mathcal{T}\}$$

Remark 13. Acceptance for the specific case of the deterministic model can be defined in this way by considering the right notion of test.

Definition 14. Let Ω be a monoid, $\mathfrak{m}$ a microcosm, $\mathcal{T}$ a test and $\mathcal{L}$ a set of languages. We say the model $\mathbb{M}^*[m, \Omega]$ *characterises the set $\mathcal{L}$ w.r.t. the test $\mathcal{T}$* if the set $\{[P]_{\mathcal{T}} \mid P \in !\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}\}$ is equal to $\mathcal{L}$.

In particular, one can show the existence of two tests $\mathcal{T}^{\mathrm{n}}$ and $\mathcal{T}^{\mathrm{co}}$ that correspond to the two notions of acceptance mentioned above and which allows for

the characterisation of usual non-deterministic classes. Those are stated in the following theorem.

Theorem 15. *The model* $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_i]$ $(i \in \mathbf{N} \cup \{\infty\})$ *characterises the class* 2NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{n}}$ and characterises the class CO2NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{co}}$.

In the case of probabilistic graphings, one can show the existence of a test $\mathscr{T}^{\mathrm{p}}$ which allows for the characterisation of probabilistic computation with unbounded error. This leads to the following theorems.

Theorem 16. *The model* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_i]$ $(i \in \mathbf{N} \cup \{\infty\})$ *characterises the class* 2PFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{p}}$.

As corollaries of these results, we obtain new characterisations of:
- the class REGULAR of Regular languages (the model $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_1]$);
- the classes NLOGSPACE and CONLOGSPACE (the model $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_\infty]$);
- the class STOCHASTIC of Stochastic languages (the model $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_1]$);
- the class PLOGSPACE (the model $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_\infty]$).

4.4. Variations

All of these results are based upon a correspondence between elements of the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ and some kinds of two-way multihead automata, either deterministic, non-deterministic or probabilistic. Several other results can be obtained by modifying the notion of automata considered. We discuss here three possible modifications.

The first modification is actually a restriction, that is: can we represent computation by *one-way automata*? One can already answer positively to this question, as the two-way capability of the automata does not really find its source in the programs P in $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ but in the representation of words. One can define an alternative representation of words over an alphabet Σ and a corresponding type $\mathbf{Words}_{\Sigma}^{(1)}$. We then obtain one-way analogues of Theorem 12, Theorem 15, and Theorem 16.

Theorem 17. *The model* $\mathbb{M}_m^{\mathrm{det}}[[0,1], \mathfrak{m}_i]$ $(i \in \mathbf{N} \cup \{\infty\})$ *characterises the class* 1DFA(i).

Theorem 18. *The model* $\mathbb{M}_m^{\mathrm{ndet}}[[0,1], \mathfrak{m}_i]$ $(i \in \mathbf{N} \cup \{\infty\})$ *characterises the class* 1NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{n}}$ and characterises the class CO1NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{co}}$.

Theorem 19. *The model* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_i]$ $(i \in \mathbf{N} \cup \{\infty\})$ *characterises the class* 1PFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{p}}$.

The second modification is the extension of automata with a *pushdown stack*. Work in this direction has recently lead to a characterisation of PTIME in a more syntactical setting [34]. Even though the syntactical approach just mentioned could very well be transported to our setting (it was shown [1] that elements of the *resolution algebra* can be represented as graphings), this would lead to a characterisation based on a microcosm containing non-measure-preserving maps. Even though non-measure-preserving maps are allowed in the general setting of graphings [1], the use of measure-preserving microcosm is more interesting in view of the possible use of mathematical invariants such as ℓ^2-Betti numbers discussed in the next section. One can find such a measure-preserving

microcosm $\mathfrak{p}$ which leads to a characterisation of PTIME in both the model of deterministic graphings and the model of non-deterministic graphings because non-determinism do not add any expressivity to the model of deterministic two-way multihead automata with a pushdown stack.

The last modification is the consideration of *quantum graphings*, i.e. graphings computing with complex numbers. This is still a work in progress, but we believe that one can define variants of quantum graphings corresponding to measure-once or measure-many quantum automata, leading to several other characterisations.

5. Complexity Constraints and Graphing Algebras

It is our belief also that important contributions can be made by using well-established mathematical techniques, tools and invariants from operator algebras and dynamical systems for addressing open problems in complexity. We now show our approach relates to well-known mathematical notions and how this enables new techniques through the computation of invariants such as ℓ^2-Betti numbers.

5.1. Compilations

The results exposed in the previous section lead us to conjecture a correspondence between complexity classes and microcosms, a conjecture that we are now able to state precisely.

It may appear that the microcosms correspond to complexity *constraints*: for instance $\mathfrak{m}_\infty$ captures the constraint of "logarithmic space computation" uniformly, i.e. the same microcosm corresponds to the same constraint for several computational paradigms (sequential, probabilistic). This correspondence, if it holds, can only be partial since, as we already explained, an extension of the microcosm $\mathfrak{m}_\infty$ which characterise the class PTIME of polynomial time predicates do not characterise NPTIME in the non-deterministic model. As a consequence, we know that all microcosms do not characterise uniformly a complexity constraint.

We will not discuss this complex question of whether complexity *constraints* can be described uniformly by a microcosm. The mere question of deciding *what is* a complexity constraint seems as difficult to answer as the question of deciding what is the right notion of algorithm. We therefore consider the already difficult question of a possible (partial) correspondence between microcosms and hierarchies of complexity classes. To formalise this, let us fix once and for all a monoid Ω, a type of graphings – e.g. probabilistic – and a test $\mathscr{T}$. In the following, we will denote by $\mathbb{C}[\mathfrak{m}]$ the set of languages characterised by the type $\mathbf{Words}_\Sigma^{(2)} \Rightarrow \mathbf{NBool}$ in the chosen model $\mathbb{M}_m^\bullet[\Omega, \mathfrak{m}]$ w.r.t. the test $\mathscr{T}$.

We now consider the following natural notion of *compilation*.

Definition 20. A measurable map $m : \mathbf{X} \to \mathbf{X}$ is *compilable* in a set of measurable maps N when there exists a finite partition $X_1, \dots, X_k$ of $\mathbf{X}$ and elements $n_1, \dots, n_k \in N$ such that $m_{\upharpoonright X_i} =_{\text{a.e.}} (n_i)_{\upharpoonright X_i}$.

Notations 21. We denoted by $m \prec_c N$ the fact that m is compilable in N. This naturally extends to a preorder on sets of measurable maps that we abusively denote by $\prec_c$ and defined as $M \prec_c N$ if and only if $m \prec_c N$ for all $m \in M$.

17

Definition 22. We define the equivalence relation on microcosms:

$$\mathfrak{m} \sim_c \mathfrak{n} \Leftrightarrow (\mathfrak{m} \prec_c \mathfrak{n} \wedge \mathfrak{n} \prec_c \mathfrak{m})$$

One can then easily show the following result, stating that – everything else being equal – two equivalent microcosms give rise to the same complexity classes.

Theorem 23. *If* $\mathfrak{m} \prec_c \mathfrak{n}$*, then* $\mathtt{C}[\mathfrak{m}] \subset \mathtt{C}[\mathfrak{n}]$.

Proof (sketch). Given a graphing G in $\mathfrak{m}$, we show how one can define a graphing $\bar{G}$ in $\mathfrak{n}$ such that for all representation of a word $W \in \mathbf{Words}^{(2)}_{\{0,1\}}$, $G\!::\!W \perp \mathscr{T}$ if and only if $\bar{G}\!::\!W \perp \mathscr{T}$. The new graphing $\bar{G}$ is defined by inductively replacing edges in G by a finite family of edges. The reason why the resulting graphing $\bar{G}$ satisfy the wanted property is comes from the fact paths between G and W are compilable in the set of paths between $\bar{G}$ and W. Consequently, cycles between $\bar{G}\!::\!W$ (the paths) and $\mathscr{T}$ used to define the orthogonality turn out to be refinements of cycles between $G\!::\!W$ and $\mathscr{T}$, which implies that $G\!::\!W \perp \mathscr{T}$ if and only if $\bar{G}\!::\!W \perp \mathscr{T}$. □

We conjecture that the converse of Theorem 23 holds, i.e. if $\mathfrak{m} \not\sim_c \mathfrak{n}$ are not equivalent, then $\mathtt{C}[\mathfrak{m}] \neq \mathtt{C}[\mathfrak{n}]$ are distinct.

Conjecture 24. *If* $\mathfrak{m} \not\sim_c \mathfrak{n}$*, then* $\mathtt{C}[\mathfrak{m}] \neq \mathtt{C}[\mathfrak{n}]$.

This would provide a partial equivalence between the problem of classifying complexity classes and that of classifying microcosms. As we will see, the first results we obtained are coherent with this conjecture. We explain below how the notion of equivalence $\sim_c$ relates (in specific cases) to well-studied notions in ergodic theory and von Neumann algebras. This fact could allow, in case the conjecture is proven, for the use of mathematical invariants as a proof method for obtaining separation results.

5.2. Measurable Equivalence Relations

Let us now explain how the microcosms, used to characterise complexity classes in the work described above, can (in some cases) generate a *measured equivalence relation*. We first recall the basics about this notion.

The notion of graphing considered in our work is in fact a generalisation of the notion considered in ergodic theory and operator algebras. The usual definition of graphings do not comprise weights. Moreover, those are usually considered built from measure-preserving maps only. We will refer to these weightless graphings as *simple graphings*.

Definition 25. A *weightless graphing* ϕ is a $\{1\}$-weighted graphing in the macrocosm $\mathfrak{M}$ over $\mathbf{X}$.

A *simple graphing* ϕ is a $\{1\}$-weighted graphing in the microcosm $\mathfrak{mp}$ of measure-preserving Borel isomorphisms over $\mathbf{X}$.

Remark 26. This definition of simple graphings is equivalent to the usual definition [45] as a family of partial measure-preserving Borel isomorphisms $\phi_f : S_f^\phi \to T_f^\phi$ where S_f^ϕ, T_f^ϕ are Borel subsets of $\mathbf{X}$.

18

Given a simple graphing, one can define an equivalence relation $\mathscr{R}(\phi)$ by considering the transitive closure of the relation

$$\{(x, \phi_f(x)) \mid f \in E^\phi, x \in S_f^\phi\}$$

This relation can be described directly in terms of paths. We define for this the set of ϕ-words as the set:

$$\text{words}(\phi) = \{f_1^{\epsilon_1} f_2^{\epsilon_2} \ldots f_k^{\epsilon_k} \mid k \geqslant 0, \forall 1 \leqslant i \leqslant k, f_i \in E^\phi, \epsilon_i \in \{-1, 1\}\}$$

Each ϕ-word $\pi = f_1^{\epsilon_1} f_2^{\epsilon_2} \ldots f_k^{\epsilon_k}$ defines a partial measure-preserving Borel isomorphism ϕ_π defined as the composite $\phi_{f_k}^{\epsilon_k} \circ \phi_{f_{k-1}}^{\epsilon_{k-1}} \circ \cdots \circ \phi_{f_1}^{\epsilon_1}$ from its domain S_π^ϕ to its codomain T_π^ϕ.

Definition 27. Let ϕ be a simple graphing. We define the equivalence relation $\mathscr{R}(\phi)$ as

$$\mathscr{R}(\phi) = \{(x, y) \mid \exists \pi \in \text{words}(\phi), x \in S_\pi^\phi, y = \phi_{pi}(x)$$

One can then show that this is a *measurable equivalence relation*, or Borel equivalence relation, i.e. an equivalence relation on X such that:
1. $\mathscr{R}(\phi)$ is a Borel subset of $\mathbf{X} \times \mathbf{X}$;
2. the set of equivalence classes of $\mathscr{R}(\phi)$ is countable;
3. every partial isomorphism $f : A \to B$ whose graph $\{(a, f(a)) \mid a \in A\}$ is a subset of $\mathscr{R}(\phi)$ is measure-preserving.

Remark 28. A microcosm $\mathfrak{m}$ is a weightless graphing. Thus every microcosm $\mathfrak{m}$ included in the microcosm $\mathfrak{mp}$ of measure-preserving Borel automorphisms gives rise to a Borel equivalence relation $\mathscr{R}(\mathfrak{m})$.

Proposition 29. *Let $\mathfrak{m}$ and $\mathfrak{n}$ be equivalent microcosms. Then $\mathscr{R}(\mathfrak{m}) = \mathscr{R}(\mathfrak{n})$.*

Borel equivalence relations were extensively studied in several domains of mathematics, as such equivalence relations are induced by measurable group actions. Indeed, if α is a measure-preserving action of a (discrete countable) group G on a (atom-free) standard Borel space $\mathbf{X}$, then one can define the Borel equivalence relation:

$$\mathscr{R}(\alpha) = \{(x, y) \mid \exists g \in G, y = \alpha(g)(x)\}$$

Conversely, it was shown by Feldman and Moore [46] that, given a measurable equivalence relation $\mathscr{R}$ on a standard Borel space $\mathbf{X}$, there exists a group G of measure-preserving Borel automorphisms of $\mathbf{X}$ such that[7] $\mathscr{R} = \mathscr{R}(G)$.

Trying to classify group actions, mathematicians have developed fine invariants to classify Borel equivalence relations. In particular, Gaboriau extended the notion of ℓ^2-Betti numbers[8] [55] to this setting, and also introduced the notion of *cost* [45] which can be understood as an approximation of the first ℓ^2-Betti number. We here recall the definition of the latter.

---

[7]In that specific case, we write G the action defined as the inclusion $G \subset \text{Aut}(\mathbf{X})$.

[8]A definition which is coherent with ℓ^2-Betti number defined by Atiyah [51], the later generalization to groups by Cheeger and Gromov [52] reformulated by Lück [53], and the generalization to von Neumann algebras by Connes and Shlyakhtenko [54].

Definition 30. The cost of a graphing ϕ is defined as the (possibly infinite) real number:

$$\mathscr{C}(\phi) = \sum_{f \in E^\phi} \lambda(S_f^\phi)$$

The cost of a measurable equivalence relation $\mathscr{R}$ is then the infimum of the cost of simple graphings generating $\mathscr{R}$:

$$\mathscr{C}(\mathscr{R}) = \inf\{\mathscr{C}(\phi) \mid \phi \text{ simple graphing s.t. } \mathscr{R} = \mathscr{R}_\phi\}$$

The cost of a group Γ is the infimum of the cost of the Borel equivalence relations $\mathscr{R}(\alpha)$ defined by free actions of Γ on a standard probability space:

$$\mathscr{C}(\Gamma) = \inf\{\mathscr{C}(\mathscr{R}(\alpha)) \mid \alpha \text{ free action of } \Gamma \text{ onto } \mathbf{X} \text{ with } \mu(X) = 1\}$$

Let us refer to Gaboriau's work for a complete overview [45], and state the result that will be of use here. A result due to Levitt [44] allows to compute the cost of finite groups, which are shown to be of fixed cost – i.e. all free actions of finite groups have the same cost. Gaboriau also computed the (fixed) cost of infinite amenable groups. The following proposition states these two results simultaneously.

Proposition 31 (Gaboriau [45])**.** *If Γ is an amenable group, then Γ is has fixed cost $1 - \frac{1}{Card(\Gamma)}$, where by convention $\frac{1}{\infty} = 0$.*

5.3. Measurable Preorders, Homotopy Equivalence

First, let us have a look at the measurable equivalence relations $\mathscr{R}(\mathfrak{m}_i)$ for $\mathfrak{m}_i$ the microcosms defined in the previous section. Notice that $\mathscr{R}(\mathfrak{m}_i) = \mathbf{Z}^2 \times \mathscr{R}(\mathfrak{s}_i)$ where $\mathscr{R}(\mathfrak{s}_i)$ is the Borel equivalence relation on $[0,1]^{\mathbf{N}}$ generated by the natural group action of $\mathfrak{G}_i$ – the group of permutations over i elements – on the Hilbert cube $[0,1]^{\mathbf{N}}$ (permutations act on the first i copies of $[0,1]$). This action being free, the Hilbert cube being a standard probability space, and the group $\mathfrak{G}_i$ being of finite order, we have that $\mathscr{C}(\mathscr{R}(\mathfrak{s}_i)) = \mathscr{R}(\mathfrak{G}_i) = 1 - \frac{1}{i!}$. This shows the following separation theorem for the microcosms $\mathfrak{m}_i$.

Theorem 32. *For all $1 \leqslant i < j \leqslant \infty$, $\mathfrak{m}_i \nsim_c \mathfrak{m}_j$.*

These results are coherent with Conjecture 24. Indeed, a number of well-known separation results such as $2\mathrm{DFA(k)} \neq 2\mathrm{DFA(k+1)}$ [56] hold for the various notion of automata considered in the previous section.

This result illustrates how one could use invariants to show that two complexity classes are distinct, under the hypothesis that Conjecture 24 is true. This uses the fact that invariants such as ℓ^2-Betti numbers can be used to show that two microcosms are not equivalent. This raises two natural questions that we answer now.

Borel equivalence relations are not enough. First the use of Borel equivalence relations is too restrictive for our purpose. Indeed, although the microcosms $\mathfrak{m}_i$ were simple graphings, we do not want to restrict the framework to those. Indeed, when one wants to represent computational principles, one sometimes want some non-invertible principles to be available. As a consequence, we are interested in microcosms that are not subsets of the microcosm $\mathfrak{m}_p$, and which are not groups.

In other words, we are not interested in group actions on a space, but rather on *monoid actions*. The problem of classifying monoid actions is however much more complex, and much less studied.

In order to have a finer analysis of the equivalence, we want to distinguish between monoids of measurable maps and groups of measurable maps. Given a weightless graphing ϕ, we consider the set of *positive ϕ-words* as the following subset of ϕ-words.

$$\mathrm{words}^+(\phi) = \{f_1 f_2 \ldots f_k \mid k \geqslant 1, \forall 1 \leqslant i \leqslant k, f_i \in E^\phi\}$$

Definition 33. Given a weightless graphing, we define the $\mathscr{P}(\phi)$ as

$$\mathscr{P}(\phi) = \{(x, y) \mid \exists \pi \in \mathrm{words}^+(\phi), y = \phi_\pi(x)\}$$

We obtain the following refinement of Proposition 29.

Proposition 34. *If two microcosms $\mathfrak{m}$ and $\mathfrak{n}$ are equivalent, the preorders $\mathscr{P}(m)$ and $\mathscr{P}(n)$ are equal.*

Can one define invariants such as ℓ^2-Betti numbers in this more general case? The notion of cost can be obviously defined for measurable preorders, but is this still an interesting invariant? In case of ℓ^2-Betti numbers, this question is quite complex as it involves the definition of a von Neumann algebra generated by a measurable preorder. Although some work defining von Neumann algebras from left-cancellable monoids could lead to a partial answer, this is a difficult open question.

Justifying the use of homotopy invariants.. The second issue is that the invariant mentioned above, namely ℓ^2-Betti numbers, are *homotopy invariants*, i.e. they are used to show that two measurable equivalence relations are *not homotopy equivalent*, which is stronger than showing that they are not equal. So, are those invariants too complex? Couldn't we use easier invariants? The answer to this question lies in a more involved study of the equivalence of microcosms. The notion of compilation we discussed above is not the finest equivalence one could consider. We chose to work with this simplified notion since the most general setting needs to go into detailed discussions about the definitions of the exponential connectives in the model. However, one should actually consider a more involved notion, namely that of *compilation up to Borel automorphisms*. This notion is not complicated to define.

Definition 35. Let Θ be a set of Borel automorphisms of $\mathbf{X}$. A measurable map m is compilable in a set N of measurable maps *up to* Θ if and only if there exists $\theta \in \Theta$, a finite partition $X_1, \ldots, X_k$ of $\mathbf{X}$ and elements $n_1, \ldots, n_k \in N$ such that $\theta \circ m_{\restriction X_i} =_{\mathrm{a.e.}} (n_i)_{\restriction X_i}$.

Then the corresponding equivalence of two microcosms $\mathfrak{m}$ and $\mathfrak{n}$ up to a set of Borel automorphisms do not induce the equality of the measurable equivalence relations $\mathscr{R}(\mathfrak{m})$ and $\mathscr{R}(\mathfrak{n})$, but only that those are homotopy equivalent. In this case, one can understand the interest of invariants such as ℓ^2-Betti numbers, or the cost which can be understood as an approximation of the first ℓ^2-Betti number.

6. Perspectives

We believe the above conjecture provides a very good working hypothesis, even if no proof of it is to be found, as the approach we propose provides an homogeneous approach to computational complexity.

This *complexity-through-realisability* theory we sketched is founded on alternative definitions of the notions of algorithms and complexity classes. The techniques were illustrated by first results showing how a large family of (predicate) complexity classes can be characterised by these techniques. Those are a first step towards a demonstration that these definitions capture and generalise standard ones, offering a unified homogeneous framework for the study of complexity classes.

Future work in this direction should therefore aim at fulfilling two main objectives. The first objective is to establish that this new approach to complexity captures, generalises and extends the techniques developed by previous logic-based approaches to computational complexity. The second objective is to establish that invariants and tools available for the mathematical theories underlying our approach can be used to address open problems in complexity, as already explained in the previous section.

6.1. A Uniform Approach to Computational Complexity

We propose the following three goals to deal with the first objective: show the *complexity-through-realisability* techniques (i) are coherent with classical theory, (ii) generalise and improve state-of-the-art techniques, and finally (iii) provide the first homogeneous theory of complexity for several computational paradigms.

Automata, Turing machines, etc.. The results presented in this paper are a first step toward showing that our technique are coherent with the classical complexity theory. However, all our results lean on previously known characterisations of complexity classes (predicates) by means of different kinds of automata. As explained above, extensions of these notions of automata can be considered to obtain further results. However, it is important to realise that our approach can also deal with other models of computation.

An adaptation of work by Asperti and Roversi [57] and Baillot [58] should allow for encoding Turing machines in some of the realisability models we consider. This should lead to characterisations of several other complexity classes (predicates), such as the exponential hierarchy. In particular this should allow for a characterisation of NPTIME, a class that – as explained above – is not characterised naturally by pushdown automata. Moreover, previous characterisations of NC_1 and PSPACE by means, respectively, of *branching programs* (Barrington's theorem [59]) and *bottleneck Turing machines* [60] should lead to characterisations of those classes.

It would also be interesting to understand if the definition of algorithms as Abstract State Machines (ASMs) proposed by Gurevich [61] corresponds to a specific case of our definition of algorithms as graphings. Although no previous work attempted to relate ASMs and GoI, an ASM is intuitively a kind of automata on first-order structures and such objects can be seen as graphings [1]. This expected result will show that the notion of graphing provides an adequate mathematical definition of the notion of computation.

Predicates, functions, etc.. All results presented above are characterisations of predicate complexity classes. It is natural to wonder if the approach is limited to those or if it applies similarly to the study of functions complexity classes.

Since one is considering models of linear logic, the approach does naturally extend to functions complexity classes. A natural lead for studying function classes is to understand the types $\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Words}_{\Sigma}^{(2)}$ – functions from (binary) natural numbers to (binary) natural numbers – in the models considered. A first step was obtained by the definition of a model of Elementary Linear Logic (ELL) [42], a system which is known to characterise elementary time functions as the set of algorithms/proofs of type $!\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Words}_{\Sigma}^{(2)}$. Although the mentioned model is not shown complete for ELL, it provides a first step in this direction as the type of functions in this model is naturally sound for elementary time functions. We believe that *complexity-through-realisability* techniques extend ICC techniques in that usual complexity classes of functions can be characterised as types of functions $!\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Words}_{\Sigma}^{(2)}$ in different models. A first natural question is that of the functions computed by elements of this type in the models described in the previous section.

Furthermore, we expect characterisations and/or definitions of complexity classes of higher-order functionals. Indeed, the models considered contain types of higher-order functionals, e.g. $(\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Words}_{\Sigma}^{(2)}) \Rightarrow (\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Words}_{\Sigma}^{(2)})$ for type 2 functionals. These models therefore naturally characterise subclasses of higher-order functionals. As no established theory of complexity for higher-order functional exists at the time, this line of research is of particular interest.

Deterministic, Probabilistic, etc.. As exposed in the previous section, the techniques apply in an homogeneous way to deterministic, non-deterministic and probabilistic automata. Moreover a generalisation towards quantum computation seems natural, as already discussed. The framework offered by graphings is indeed particularly fit for both the quantum and the probabilistic computational paradigm as it is related to operator algebras techniques, hence to both measure theory and linear algebra. In this particular case of probabilistic and quantum computation, we believe that it will allow for characterisations of other standard probabilistic and quantum complexity classes such as PPTIME, BPPTIME, or BQPTIME.

Furthermore, the generality of the approach and the flexibility of the notion of graphing let us hope for application to other computational paradigms. Among those, we can cite concurrent computation and cellular automata. On one hand, this would provide a viable and well-grounded foundation for a theory of computational complexity for concurrent computation, something currently lacking. On the other hand, applying the techniques to cellular automata will raise the question of the relationship between "classical" complexity theory and the notion of *communication complexity*.

6.2. Using Mathematical Invariants

As we already discussed in the previous section, the proposed framework has close ties with some well-studied notions from other fields of mathematics. Of course, the possibility of using invariants from these fields in order to obtain new separation results would be a major step for complexity theory. However, let us notice that even though nothing forbids such results, the proposed method do not provide a miraculous solution to long-standing open problems. Indeed, in

order to obtain separation results using the techniques mentioned in the previous sections, one would have to

- first prove Conjecture 24, which is not the simplest thing to do;
- then characterise the two classes to be separated by models $\mathbb{M}_m^\bullet[\Omega, \mathfrak{m}]$ and $\mathbb{M}_m^\bullet[\Omega, \mathfrak{m}]$ which differ only by the chosen microcosm (either $\mathfrak{m}$ or $\mathfrak{n}$);
- then compute invariants showing that these microcosms are not homotopy equivalent (if the microcosms are groups of measure-preserving Borel isomorphisms), or – in the general case – define and then compute homotopy invariants separating the two microcosms.

As an illustration, let us consider what could be done if someone were to find a proof of Conjecture 24 today. Based on the results exposed above, one could only hope for already known separation results, namely the fact that the inclusions 2DFA(k) $\subsetneq$ 2DFA(k+1), 2NFA(k) $\subsetneq$ 2NFA(k+1), CO2NFA(k) $\subsetneq$ CO2NFA(k+1) and 2PFA(k) $\subsetneq$ 2PFA(k+1). Those, as explained above, would be obtained by computing the cost of the associated Borel equivalence relations.

If one were to consider the characterisation of PTIME obtained syntactically [34] as well, one could hope for separating the classes LOGSPACE and PTIME. However, the microcosm used to characterise PTIME is neither a group nor consists in measure-preserving maps, two distinct but major handicaps for defining and computing invariants. As explained, future work will show that one can actually consider a microcosm of measure-preserving maps to characterise PTIME. But once again, this microcosm would not be a group, hence one would need to extends the theory of invariants for Borel equivalence relations in order to hope for a separation result. Last, but not least, these invariants should be expressive enough! For instance, the cost of the relation induced by the microcosm $\mathfrak{m}_\infty$ is equal to 1, as is the cost of every amenable infinite group. Amenable extensions of this microcosm will henceforth not be shown to be strictly greater than $\mathfrak{m}_\infty$ unless some other, finer, invariants are considered and successfully computed!

However, the proposed proof method does not naturally appear as a *natural proof* in the sense of Razborov and Rudich [11]. It is the author's opinion that this simple fact on its own makes the approach worth studying further.

References

References

[1] T. Seiller, Interaction Graphs: Graphings, Submitted .

[2] C. Aubert, T. Seiller, Characterizing co-NL by a group action, Mathematical Structures in Computer Science doi:0.1017/S0960129514000267, to appear.

[3] C. Aubert, T. Seiller, Logarithmic Space and Permutations, Information and Computation To appear.

[4] T. Seiller, Interaction Graphs and Complexity I, in preparation., 2015.

[5] J. Hartmanis, R. Stearns, On the computational complexity of algorithms, Transactions of the American Mathematical Society 117.

[6] N. Immerman, Nondeterministic space is closed under complementation, in: Structure in Complexity Theory Conference, 1988.

[7] A. Cobham, The intrinsic computational difficulty of functions, in: Proceedings of the 1964 CLMPS, 1965.

[8] S. Cook, The complexity of theorem-proving procedures, in: Proceedings of the 3rd ACM Symposium on Theory of Computing, 1971.

[9] J. Hartmanis, R. Stearns, On the computational complexity of algorithms, Transactions of the AMS 117.

[10] W. Savitch, Relationship between nondeterministic and deterministic tape classes, Journal of Computer and Systems Sciences 4.

[11] A. A. Razborov, S. Rudich, Natural proofs, Journal of Computer and System Sciences 55.

[12] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: SIAM-AMS Proc., vol. 7, 1974.

[13] N. Jones, A. Selman, Turing machines and the spectra of first-order formulas, Journal of Symbolic Logic 39.

[14] K. J. Compton, E. Grädel, Logical definability of counting functions, J. Comput. Syst. Sci. 53.

[15] A. Dawar, E. Grädel, The descriptive complexity of parity games, LMCS 5213.

[16] E. Grädel, Y. Gurevich, Tailoring recursion for complexity, Journal of Symbolic Logic 60.

[17] R. Szelepscényi, The method of forced enumeration for nondeterministic automata, Bulletin of the EATCS 33.

[18] S. Bellantoni, S. Cook, A new recursion-theoretic characterization of the polytime functions, Computational Complexity 2.

[19] D. Leivant, J.-Y. Marion, Lambda calculus characterizations of poly-time, Fundam. Inform. 19.

[20] D. Leivant, J.-Y. Marion, Ramified recurrence and computational complexity II: Substitution and poly-space, Lecture Notes in Computer Science 933.

[21] J.-Y. Girard, Linear logic, Theoretical Computer Science 50 (1) (1987) 1–101, doi:10.1016/0304-3975(87)90045-4.

[22] J.-Y. Girard, A. Scedrov, P. J. Scott, Bounded linear logic: a modular approach to polynomial-time computability, Theor. Comput. Sci. 97 (1) (1992) 1–66, ISSN 0304-3975, doi:10.1016/0304-3975(92)90386-T, URL http://dx.doi.org/10.1016/0304-3975(92)90386-T.

[23] V. Danos, J.-B. Joinet, Linear Logic & Elementary Time, Information and Computation 183 (1) (2003) 123–137, doi:10.1016/S0890-5401(03)00010-5.

[24] J.-Y. Girard, Towards a geometry of interaction, in: Proceedings of the AMS Conference on Categories, Logic and Computer Science, 1989.

[25] S. C. Kleene, On the interpretation of intuitionistic number theory, Journal of Symbolic Logic 10.

[26] J.-L. Krivine, Typed lambda-calculus in classical Zermelo-Fraenkel set theory, Arch. Mathematical Logic 40.

[27] J.-Y. Girard, Geometry of Interaction I: Interpretation of System F, in: In Proc. Logic Colloquium 88, 1989.

[28] G. Gonthier, M. Abadi, J.-J. Lévy, The Geometry of Optimal Lambda Reduction, in: R. Sethi (Ed.), POPL, ACM Press, ISBN 0-89791-453-8, 15–26, doi:10.1145/143165.143172, 1992.

[29] J. Lamping, An Algorithm for Optimal Lambda Calculus Reduction, in: F. E. Allen (Ed.), POPL, ACM Press, ISBN 0-89791-343-4, 16–30, doi:10.1145/96709.96711, 1990.

[30] P. Baillot, M. Pedicini, Elementary complexity and geometry of interaction, Fundamenta Informaticae 45 (1-2) (2001) 1–31.

[31] U. D. Lago, The Geometry of Linear Higher-order Recursion, ACM Trans. Comput. Logic 10 (2) (2009) 8:1–8:38, ISSN 1529-3785, doi:10.1145/1462179.1462180, URL http://doi.acm.org/10.1145/1462179.1462180.

[32] J.-Y. Girard, Normativity in Logic, in: P. Dybjer, S. Lindström, E. Palmgren, G. Sundholm (Eds.), Epistemology versus Ontology, vol. 27 of *Logic, Epistemology, and the Unity of Science*, Springer, ISBN 978-94-007-4434-9, 978-94-007-4435-6, 243–263, 2012.

[33] C. Aubert, M. Bagnol, P. Pistone, T. Seiller, Logic Programming and Logarithmic Space, in: J. Garrigue (Ed.), Programming Languages and Systems - 12th Asian Symposium, APLAS 2014, Singapore, November 17-19, 2014, Proceedings, vol. 8858 of *Lecture Notes in Computer Science*, Springer, ISBN 978-3-319-12735-4, 39–57, doi:10.1007/978-3-319-12736-1_3, URL http://dx.doi.org/10.1007/978-3-319-12736-1_3, 2014.

[34] C. Aubert, M. Bagnol, T. Seiller, Memorization for Unary Logic Programming: Characterizing Ptime, Submitted .

[35] J.-Y. Girard, Geometry of Interaction IV: the Feedback Equation, in: Stoltenberg-Hansen, Väänänen (Eds.), Logic Colloquium '03, 76–117, 2006.

[36] J.-Y. Girard, Geometry Of Interaction III: Accommodating The Additives, in: Advances in Linear Logic, no. 222 in Lecture Notes Series, Cambridge University Press, 329–389, 1995.

[37] T. Seiller, Logique dans le facteur hyperfini : géometrie de l'interaction et complexité, Ph.D. thesis, Université Aix-Marseille, URL http://tel.archives-ouvertes.fr/tel-00768403/, 2012.

[38] T. Seiller, A Correspondence between Maximal Abelian Sub-Algebras and Linear Logic Fragments, Submitted .

[39] A. Sinclair, R. Smith, Finite von Neumann algebras and Masas, no. 351 in London Mathematical Society Lecture Note Series, Cambridge University Press, 2008.

[40] T. Seiller, Interaction Graphs: Multiplicatives, Annals of Pure and Applied Logic 163 (2012) 1808–1837, doi:10.1016/j.apal.2012.04.005.

[41] T. Seiller, Interaction Graphs: Additives, Accepted for publication in Annals of Pure and Applied Logic .

[42] T. Seiller, Interaction Graphs: Exponentials, Submitted .

[43] S. Adams, Trees and amenable equivalence relations, Ergodic Theory and Dynamical Systems 10 (1990) 1–14.

[44] G. Levitt, On the cost of generating an equivalence relation, Ergodic Theory and Dynamical Systems 15 (1995) 1173–1181, doi:10.1017/S0143385700009846.

[45] D. Gaboriau, Coût des relations d'équivalence et des groupes, Inventiones Mathematicae 139 (2000) 41–98.

[46] J. Feldman, C. C. Moore, Ergodic equivalence relations, cohomology, and von Neumann algebras. I, Transactions of the American mathematical society 234 (2) (1977) 289–324.

[47] B. Fuglede, R. V. Kadison, Determinant theory in finite factors, Annals of Mathematics 56 (2).

[48] J.-Y. Girard, Geometry of Interaction V: Logic in the Hyperfinite Factor., Theoretical Computer Science 412 (2011) 1860–1883.

[49] T. Seiller, Interaction Graphs and Complexity, extended Abstract, 2014.

[50] V. Danos, La Logique Linéaire Appliquée à l'Étude de Divers Processus de Normalisation (principalement du λ-calcul), Ph.D. thesis, Paris VII University, 1990.

[51] M. Atiyah, Elliptic operators, discrete groups and von Neumann algebras, vol. 32-33 of *Astérisque*, Société Mathématique de France, 43–72, 1976.

[52] J. Cheeger, M. Gromov, L^2-Cohomology and group cohomology, Topology 25 (2) (1986) 189–215.

[53] W. Lück, L^2-Invariants: Theory and Applications to Geometry and K-Theory, vol. 44 of *A Series of Modern Surveys in Mathematics*, 2002.

[54] A. Connes, D. Shlyakhtenko, L 2-homology for von Neumann algebras, Journal für die reine und angewandte Mathematik 2005 (586) (2005) 125–168.

[55] D. Gaboriau, Invariants l2 de relations d'équivalence et de groupes, Publ. Math. Inst. Hautes Études Sci 95 (93-150) (2002) 15–28.

[56] B. Monien, Transformational Methods and their Application to Complexity Problems, Acta Informatica 6 (1976) 95–108.

[57] A. Asperti, L. Roversi, Intuitionistic light affine logic, ACM Transactions on Computational Logic (TOCL) 3 (1) (2002) 137–175.

[58] P. Baillot, Elementary Linear Logic Revisited for Polynomial Time and an Exponential Time Hierarchy, in: H. Yang (Ed.), APLAS, vol. 7078 of *Lecture Notes in Computer Science*, Springer, ISBN 978-3-642-25317-1, 337–352, 2011.

[59] D. A. Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in {NC1}, Journal of Computer and System Sciences 38 (1) (1989) 150 − 164.

[60] J.-Y. Cai, M. Furst, PSPACE Survives Constant-With Bottlenecks, International Journal of Foundations of Computer Science 02 (01) (1991) 67–76.

[61] Y. Gurevich, Specification and Validation Methods, chap. Evolving Algebras 1993: Lipari Guide, Oxford University Press, Inc., New York, NY, USA, ISBN 0-19-853854-5, 9–36, 1995.